ABDULLAH GÜL
ÜNİVERSİTESİ

SAPA CAPSULE PROJECT
ELECTRONIC STETHOSCOPE

PROJECT TEAM

MURAT CAN MUTLU
OSMAN SAMET ŞENOL

Table of Contents

Chapter 1

Introduction

1.1) OBJECTIVE

The primary objective of this capsule project is to conceptualize and construct an electronic stethoscope system with the explicit goal of capturing acoustic manifestations emanating from diverse anatomical regions and subsequently transducing these auditory inputs into digital signals. Furthermore, the analytical dimension of this project involves the transformation of analog signals acquired from the electronic stethoscope into digital counterparts through the utilization of MATLAB. This analytical phase seeks to meticulously scrutinize the unprocessed and filtered manifestations of the signals in both the frequency and time domains.

1.2) BACKGROUND

The stethoscope, recognized as a medical apparatus, is frequently employed by healthcare practitioners such as physicians and nurses. Its principal purpose lies in its capacity to discern auditory phenomena emanating from the human body. Structurally, it comprises two auditory receivers and a thoracic component. Ever since the advent of the initial stethoscope by René Laennec in 1816, medical practitioners have consistently worn stethoscopes with assurance and reliance (Bilberg et all.,2008).

Electronic stethoscopes present numerous benefits in comparison to traditional acoustic counterparts, encompassing features such as noise attenuation, augmented amplification, and the capability to both store and transmit auditory data (Rennoll et all.,2020). The use of stethoscopes, which are simple acoustic devices that enable medical professionals to examine sounds produced by the body, is complemented by their digital counterparts. Digital stethoscopes, while keeping the basic diagnostic function, have various features and offer practical benefits in clinical settings.

The main advantage of these instruments is their ability to filter sounds by frequency. These devices can enhance or selectively filter sounds within specific frequency ranges, allowing medical practitioners to focus on certain physiological sounds and conduct a detailed examination (Rennoll et all.,2020). Moreover, these instruments demonstrate the capability for audio recording, a functionality that expedites a thorough examination of the patient's physiological sounds (Mallik et all.,2017). Such recorded audio not only enhances the granularity of examinations but also serves as a valuable point of reference within the realms of diagnostic and therapeutic protocols. Additionally, digital stethoscopes incorporate sound amplification features, augmenting the perceptibility of subdued or distant sounds, thereby empowering healthcare professionals to conduct a more precise evaluation.

Chapter 2
Analytical And Simulation Procedures

Components	Numbers
1.5uF capacitor	3
68nF capacitor	1
220nF capacitor	1
150nF capacitor	3
10nF capacitor	1
47nF capacitor	1
10K	2
1K	2
8.2K	2
3.3K	1
4.7K	2
6.8K	1
2.7K	1
15K	1
3.9K	2
TL072CP Opamp	3
LM358P	2
3.5mm Jack	1
Gate Switch	2
Arduino Uno	1

Table1.1:

In this project, 1.5uF(3), 8.2K, 3.3K components were used for 20 Hz high pass and 4.7K (2), 8.2K, 68nF, 220nF components were used for 200 Hz low pass in the circuit established to capture pure heart sound. Also, 150nF(3), 6.8K, 2.7K components were used for 200 Hz high pass and 3.9K(2), 15K, 10nF, 47nF components were used for 1000 Hz low pass in the circuit established to capture lung sound. Also, 1K(2), 10K(2) resistor was used to amplify the received signal and the total gain was equal to 100 times. Filters and amplifiers are connected to each other using buffers. Therefore, all filter topologies are based on Multiple feedback Filter and filters circuit were designed and its schematics were drawing on LTSpice simulation software that is circuit drawing program based on Spice. Additionally, Arduino UNO was used to convert the analog signal coming from the circuit output into a digital signal and to transmit this data to the MATLAB software and plot the received data. In addition, due to the presence of negative responses in the received analog signal, the voltage range that the analog pin of the Arduino can read is between 0-5V, so negative 2.25 VDC voltage was placed in the Arduino's range by giving it to the ground of the Arduino. Finally, thanks to the data transferred with Arduino, Matlab's AppDesigner application, the cut-off frequencies can be adjusted, and Gui was designed, which plots the captured image, the filtered image and their ffts, that is, the frequency response.

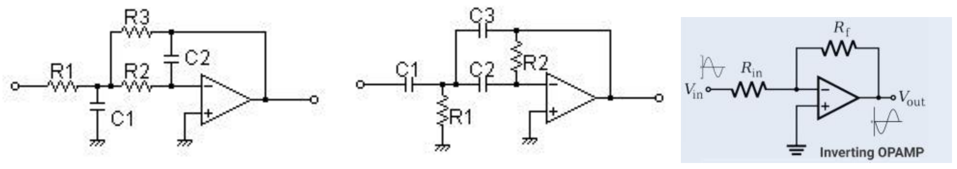

Figure.1: Low pass filter, high pass filter and inverting amplifier topologies respectively.

The transfer function is a mathematical function that allows us to understand how systems or circuits respond to which frequencies.

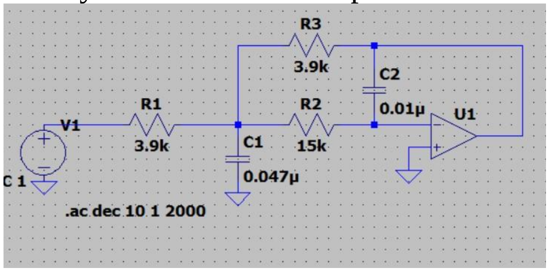

Figure.2: Low pass filter that has 1000 Hz cut off frequency

Transfer function of 1000Hz Low-pass Filter:

$H(s) = -\frac{36370249.136207}{s^2+12329.514457174s+36370249.136207}$ so, Cut-off frequency: $f_c \approx 1000$ Hz

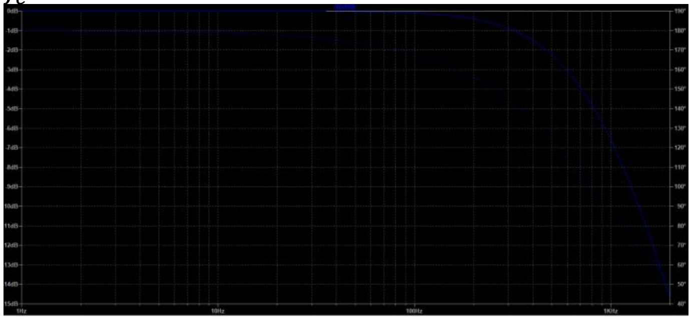

Figure.3: Bode plot off 1000 Hz low pass filter.

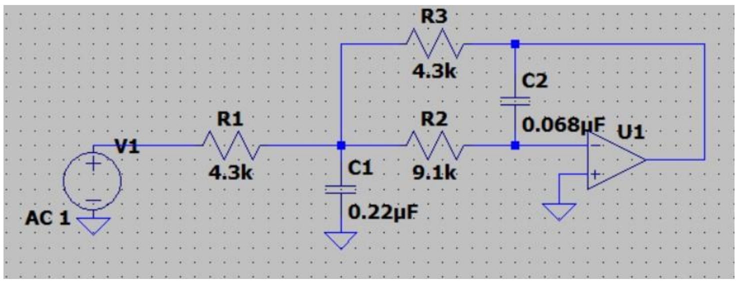

Figure.4: Low pass filter that has 200 Hz cut off frequency.

Transfer function of 200Hz Low-pass Filter:

$H(s) = -\frac{1734429.6778956}{s^2+2488.5597018446s+1734429.6778956}$ so, Cut-off frequency: $f_c \approx 200$ Hz

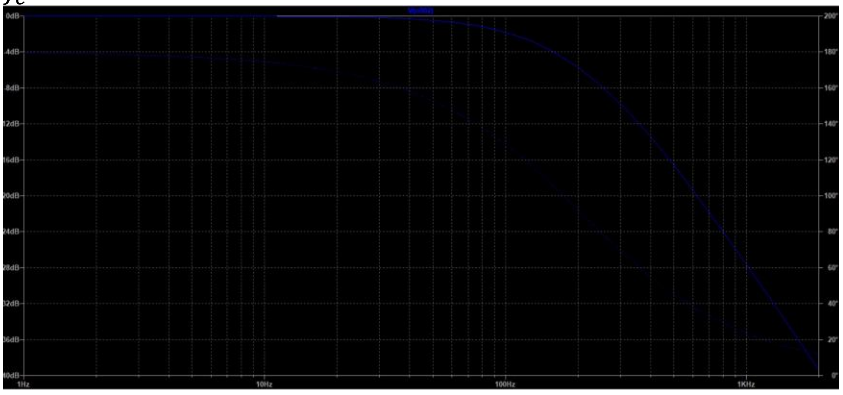

Figure.5: Bode plot off 200 Hz low pass filter.

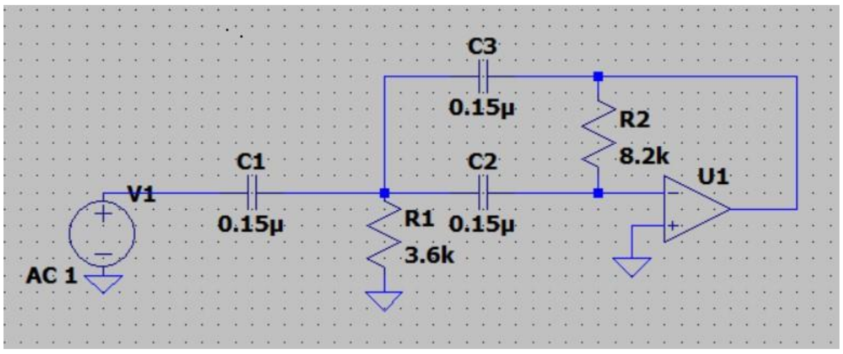

Figure.6: High pass filter that has 200 Hz cut off frequency.

Transfer function of 200Hz high-pass Filter:

$H(s) = -\frac{s^2}{s^2+2439.0243902439s+1642440.6668309}$ so, Cut-off frequency: $f_c \approx 200$ Hz

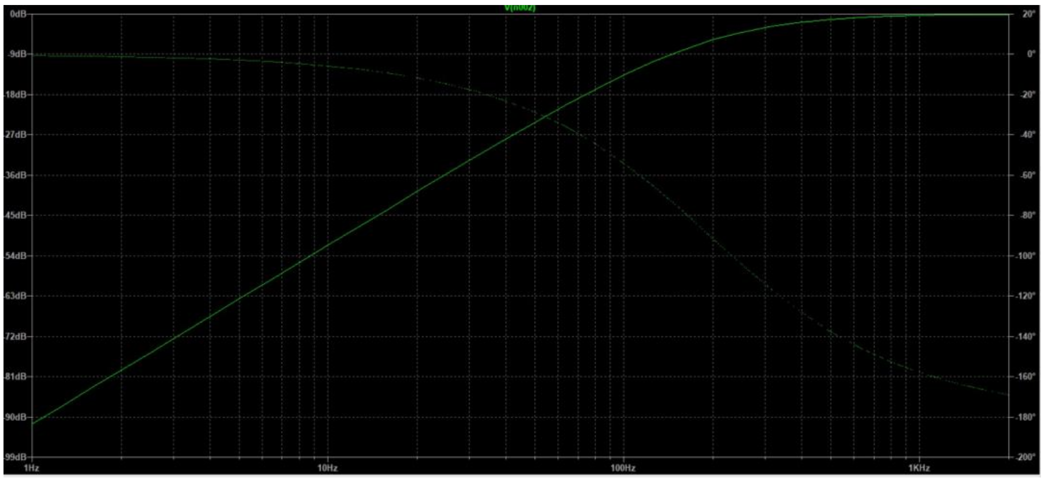

Figure.7: Bode plot off 200 Hz high pass filter.

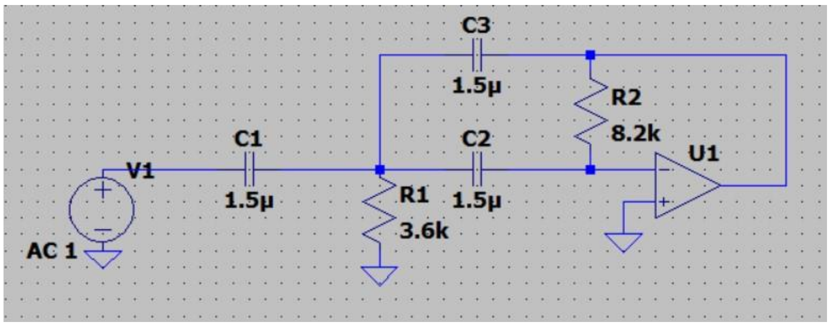

Figure.8: High pass filter that has 20 Hz cut off frequency.

Transfer function of 20Hz high-pass Filter:

$H(s) = -\frac{s^2}{s^2+243.90243902439s+16424.406668309}$ so, Cut-off frequency: $f_c \approx 20$ Hz

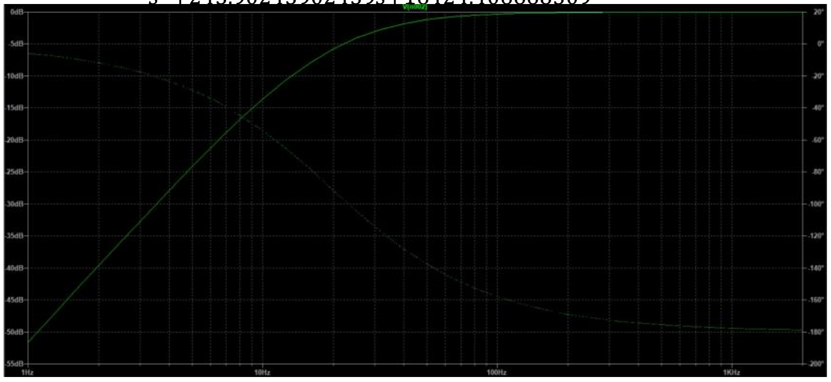

Figure.9: Bode plot off 20 Hz high pass filter.

The main circuit is shown in the Figure 10.

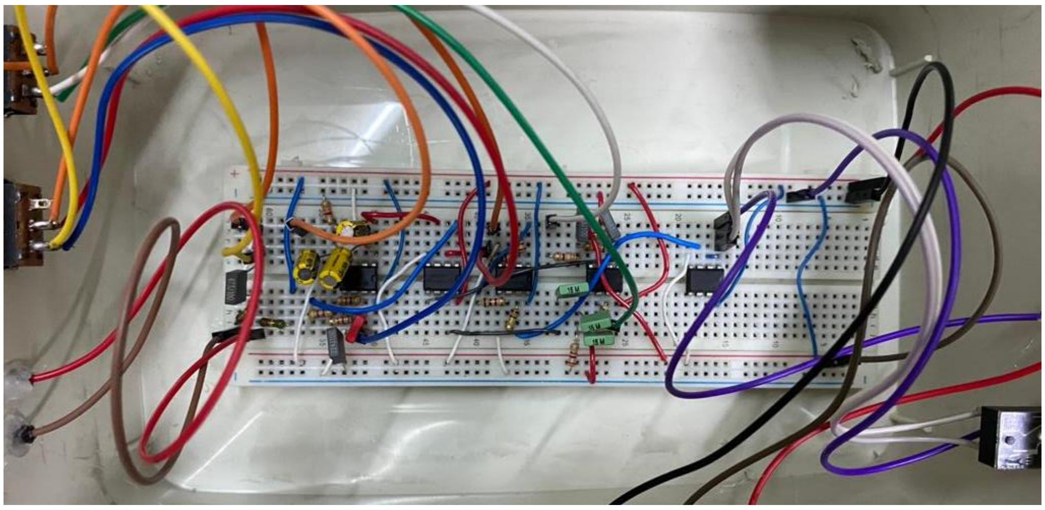

Figure.10: The main circuit.

The filter for hearth sound is shown in Figure 11.

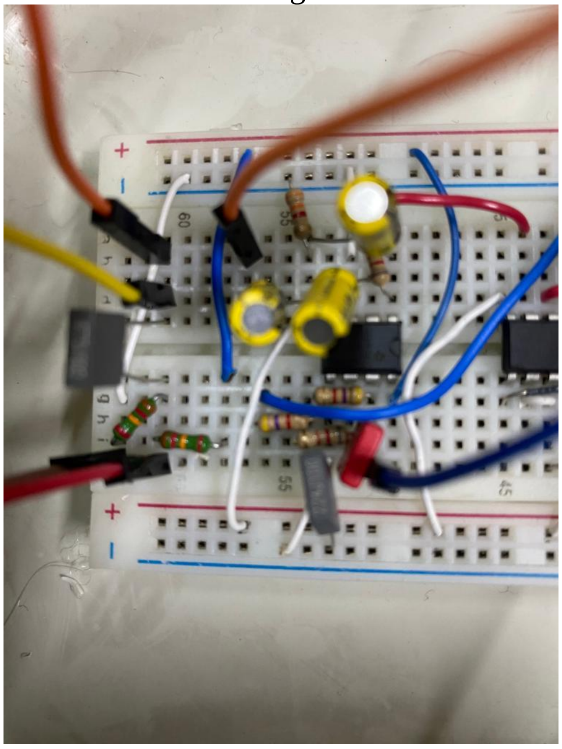

Figure.11: The filter for hearth sound.

The filter for lung sound is shown in Figure 12.

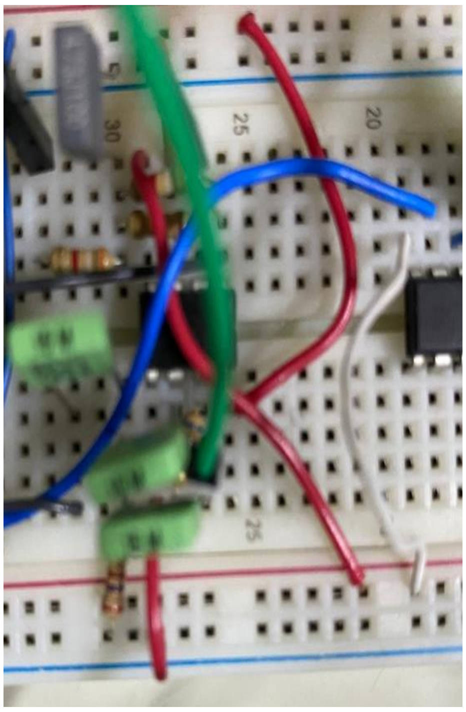
Figure.12: The filter for lung sound.

The amplifier circuit is shown in Figure 13.

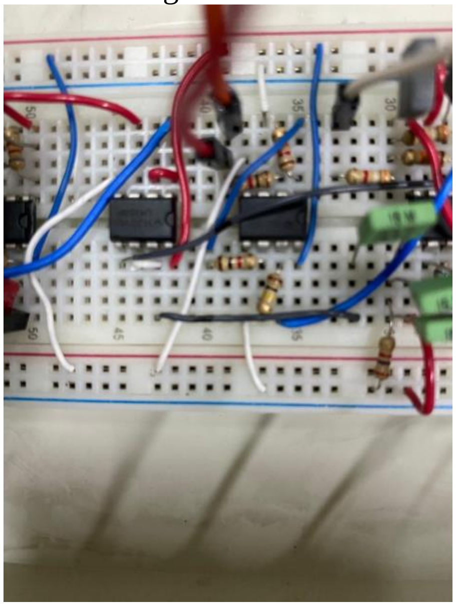
Figure.13: The amplifier circuit.

Chapter 3

Results and Conclusion

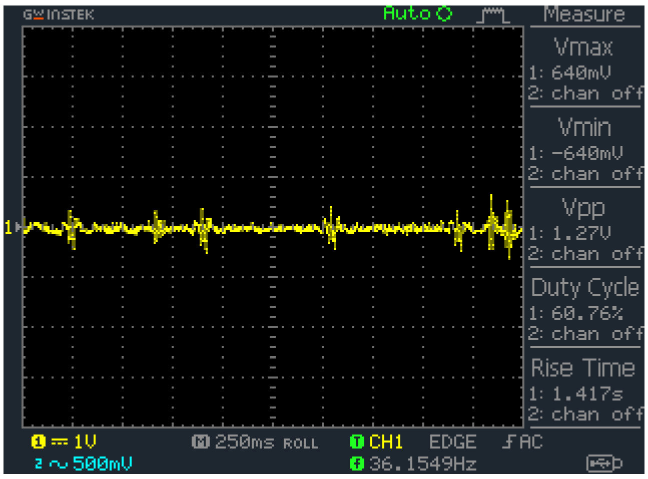

Figure.10: The signal of hearth on oscilloscope.

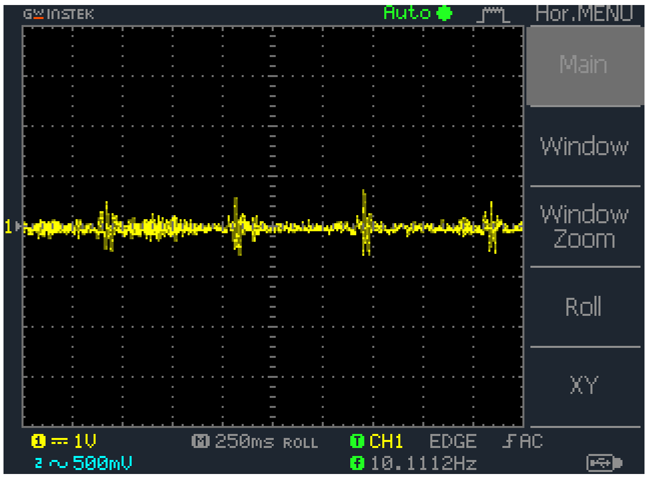

Figure.11: The signal of respiratory on oscilloscope.

The circuit employed a multi-feedback filter configuration. The primary rationale underlying the utilization of the aforementioned filter configuration within the circuit lies in its enhanced efficacy in attenuating undesired frequencies with a heightened degree of precision and rapidity, in stark contrast to both Sallen Key and Chebyshev filter counterparts. The discernible superiority of this particular filter design emanates from its ability to exhibit a more pronounced and expeditious reduction in signal amplitudes corresponding to frequencies outside the designated passband, thereby affording a more robust and selective means of signal conditioning. In comparison to its Sallen Key counterpart, characterized by relatively gentler frequency response characteristics, and the Chebyshev filter, which exhibits passband ripple at the expense of steeper roll-off rates, the selected filter configuration excels in achieving a more decisive delineation between desired and undesirable frequency components. Consequently, the judicious deployment of this filter type is instrumental in fostering a heightened precision in frequency response shaping within the circuit, thereby optimizing its performance and ensuring a more effective filtration of unwanted signal components (Tran et all,.2020).

As a consequence of the conducted experiments, the apparatus successfully attenuated extraneous environmental acoustic stimuli. Moreover, it underwent systematic calibration in accordance with the specific frequencies corresponding to the targeted physiological organs, thereby facilitating the discernment and audibility of their respective acoustic manifestations. This empirical endeavor culminated in the efficacious mitigation of ambient auditory interference, coupled with a refined optimization of the apparatus's capacity to amplify and elucidate the distinctive acoustic signatures emanating from the specified anatomical structures.

Through the employment of signal processing techniques within the MATLAB software environment, the analog data emanating from the circuit underwent a systematic transformation into digital form. Subsequently, comprehensive analyses were conducted, encompassing the acquisition of raw data, various

filtered iterations, as well as graphical representations delineating both the raw signal within the frequency domain and the filtered signal within the frequency domain. These multifaceted outcomes were accomplished through the utilization of the App Designer tool, contributing to a nuanced and in-depth exploration of the signal characteristics and processing nuances inherent in the experimental framework.
The data that was obtained from the MATLAB is shown in Figure 14.

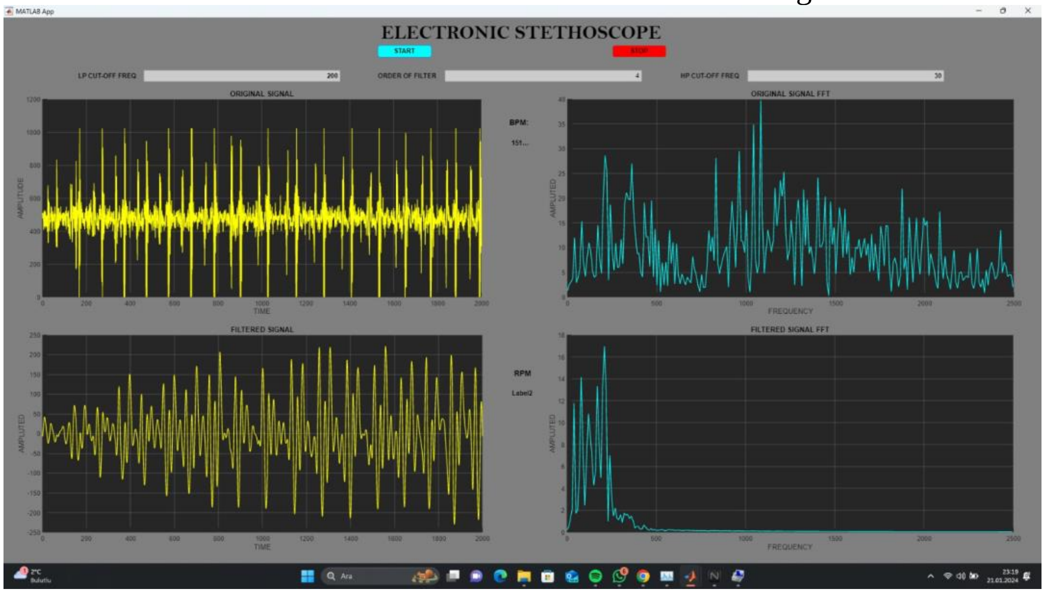

Figure.14: The data that was obtained from MATLAB.

3.1)MATLAB SIGNAL PROCESSING CODES

```
classdef GUI < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                     matlab.ui.Figure
        ELECTRONICSTETHOSCOPELabel   matlab.ui.control.Label
        Label2                       matlab.ui.control.Label
        RPMLabel                     matlab.ui.control.Label
        BPMLabel                     matlab.ui.control.Label
        LableLabel                   matlab.ui.control.Label
        ORDEROFFILTEREditField
matlab.ui.control.NumericEditField
        ORDEROFFILTEREditFieldLabel  matlab.ui.control.Label
        LPCUTOFFFREQEditField
matlab.ui.control.NumericEditField
        LPCUTOFFFREQEditFieldLabel   matlab.ui.control.Label
        HPCUTOFFFREQEditField
matlab.ui.control.NumericEditField
        HPCUTOFFFREQEditFieldLabel   matlab.ui.control.Label
        STOPButton                   matlab.ui.control.Button
        STARTButton                  matlab.ui.control.Button
        UIAxes_4                       matlab.ui.control.UIAxes
        UIAxes_3                       matlab.ui.control.UIAxes
        UIAxes_2                       matlab.ui.control.UIAxes
        UIAxes                         matlab.ui.control.UIAxes
    end


     properties (Access = private)
        value
        f
        f1
        cf1
        cf2
        X1
        X2
        X3
        X4
        serialport
        Fs = 8000;
        fs = 5000;
        fc1 = 30;
        fc2 = 200;
        T = 1/5000;
        m = 1;
        plotWindow = 2000;
        IsRunning = false;
        Timer
        Data
        ffData
        fdata
        coefficient_1
        coefficient_2
        fc3=35
        bpm
        coefficient_3
        coefficient_4
        orderOfFilter=4;
        end

    methods (Access = private)

        function timerCallback = func(app)
            if ~app.IsRunning
            stop(app.Timer);
        end
    end
end

    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            app.IsRunning=true;
            app.Timer=timer;
            app.Timer.Period=0.1;
            app.Timer.ExecutionMode='fixedRate';
            app.Timer.TimerFcn=@(~,~)app.timerCallback;
            start(app.Timer);


        end

        % Button pushed function: STARTButton
        function STARTButtonPushed(app, event)
            app.IsRunning = true;
            app.value = serialport("COM8", 9600);
            configureTerminator(app.value, "CR/LF");
            flush(app.value);
            app.value.UserData = struct("Data", [], "Order", 1, 
"fData", [],"ffData",[]);
            app.IsRunning = true;

            while app.value.UserData.Order < 100000
                    app.Data = readline(app.value);
                    app.value.UserData.Data(end+1) = 
str2double(app.Data);
                    app.value.UserData.Order = 
app.value.UserData.Order + 1;
                if mod(app.value.UserData.Order, 10) == 0
                    configureCallback(app.value, "off");

                    plot(app.UIAxes, 
app.value.UserData.Data(max(1,end-app.plotWindow+1):end),'Color','y')
                    [app.coefficient_1, app.coefficient_2] = butter(4, 
app.fc3/(app.fs/2), 'high');
                    app.value.UserData.ffData = 
filter(app.coefficient_1, app.coefficient_2, 
app.value.UserData.Data(1:end));
                    [app.coefficient_3, app.coefficient_4] = 
butter(app.orderOfFilter, [app.fc1/(app.fs/2), app.fc2/(app.fs/2)], 
'bandpass');

                    app.value.UserData.fData = 
filter(app.coefficient_3, app.coefficient_4, app.value.UserData.Data);

                    plot(app.UIAxes_2, app.value.UserData.fData(max(1, 
end-app.plotWindow+1):end) * 3,'Color','y');
                    drawnow;

                    configureTerminator(app.value, "CR/LF");

                end
                 if mod(app.value.UserData.Order,500)==0
                     configureCallback(app.value, "off");

                     app.cf1 = fft(app.value.UserData.ffData(end-
499+1:end));
                     app.X2 = abs(app.cf1 / 500); % Use 400 for 
normalization
```

```
                     app.f = app.fs *
(0:(length(app.cf1)/2))/length(app.cf1); % Adjust frequency axis
                     app.X1 = app.X2(1:length(app.cf1)/2+1);
                     app.X1(2:end-1) = 2 * app.X1(2:end-1);
                     plot(app.UIAxes_3,app.f,app.X1,'Color','c');
                     set(gca,'ylim',[0,10]);

                     app.cf2 = fft(app.value.UserData.fData(end-
499+1:end));
                     app.X3 = abs(app.cf2 / 500); % Use 400 for
normalization
                     app.f1 = app.fs *
(0:(length(app.cf2)/2))/length(app.cf2); % Adjust frequency axis
                     app.X4 = app.X3(1:length(app.cf2)/2+1);
                     app.X4(2:end-1) = 2 * app.X4(2:end-1);
                     plot(app.UIAxes_4, app.f1,app.X4,'Color','c');
                     set(gca,'ylim',[0,15]);

                     grid on;
                     app.m=app.m+500;
                     configureTerminator(app.value,"CR/LF");
                 end
                   if mod(app.value.UserData.Order, 1500) == 0
                     % Measure elapsed time
                    timeAtPointBPM = toc;
                    % Reset the timer immediately for the next
interval


                    % Detect peaks in the data
                    threshold = 490; % Adjust this threshold as needed
                    [peaks, ~] =
findpeaks(app.value.UserData.Data(end-1499+1:end), 'MinPeakHeight',
threshold);
                    numPeaks = numel(peaks); % Count the number of
peaks

                    % Calculate BPM
                    app.bpm = (60 * numPeaks) / timeAtPointBPM; % BPM
calculation
                    disp(int16(app.bpm)); % Display BPM
                    tic;
                    app.LableLabel.Text = num2str(app.bpm);
                    end
                end
        end

        % Button pushed function: STOPButton
        function STOPButtonPushed(app, event)
            app.IsRunning=false;
            delete(app.value);
        end

        % Value changed function: LPCUTOFFFREQEditField
        function LPCUTOFFFREQEditFieldValueChanged(app, event)
```

```
            b=app.LPCUTOFFFREQEditField.Value;
            app.fc2=b;
        end

        % Value changed function: HPCUTOFFFREQEditField
        function HPCUTOFFFREQEditFieldValueChanged(app, event)
             c= app.HPCUTOFFFREQEditField.Value;
             app.fc1=c;
        end

        % Value changed function: ORDEROFFILTEREditField
        function ORDEROFFILTEREditFieldValueChanged(app, event)
            j = app.ORDEROFFILTEREditField.Value;
            app.orderOfFilter=j;
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are
created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Color = [0.502 0.502 0.502];
            app.UIFigure.Position = [100 100 1127 792];
            app.UIFigure.Name = 'MATLAB App';

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'ORIGINAL SIGNAL')
            xlabel(app.UIAxes, 'TIME')
            ylabel(app.UIAxes, 'AMPLITUDE')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Color = [0.149 0.149 0.149];
            app.UIAxes.GridColor = [1 1 1];
            app.UIAxes.MinorGridColor = [0 0 0];
            app.UIAxes.XGrid = 'on';
            app.UIAxes.YGrid = 'on';
            app.UIAxes.ZGrid = 'on';
            app.UIAxes.Position = [24 350 478 311];

            % Create UIAxes_2
            app.UIAxes_2 = uiaxes(app.UIFigure);
            title(app.UIAxes_2, 'FILTERED SIGNAL')
            xlabel(app.UIAxes_2, 'TIME')
            ylabel(app.UIAxes_2, 'AMPLUTED')
            zlabel(app.UIAxes_2, 'Z')
            app.UIAxes_2.Color = [0.149 0.149 0.149];
            app.UIAxes_2.GridColor = [1 1 1];
            app.UIAxes_2.XGrid = 'on';
            app.UIAxes_2.YGrid = 'on';
            app.UIAxes_2.ZGrid = 'on';
            app.UIAxes_2.Position = [26 23 478 311];

            % Create UIAxes_3
            app.UIAxes_3 = uiaxes(app.UIFigure);
            title(app.UIAxes_3, 'ORIGINAL SIGNAL FFT')
            xlabel(app.UIAxes_3, 'FREQUENCY')
            ylabel(app.UIAxes_3, 'AMPLUTED')
            zlabel(app.UIAxes_3, 'Z')
            app.UIAxes_3.Color = [0.149 0.149 0.149];
            app.UIAxes_3.GridColor = [1 1 1];
            app.UIAxes_3.XGrid = 'on';
            app.UIAxes_3.YGrid = 'on';
            app.UIAxes_3.ZGrid = 'on';
            app.UIAxes_3.Position = [612 350 478 311];

            % Create UIAxes_4
            app.UIAxes_4 = uiaxes(app.UIFigure);
            title(app.UIAxes_4, 'FILTERED SIGNAL FFT')
            xlabel(app.UIAxes_4, 'FREQUENCY')
            ylabel(app.UIAxes_4, 'AMPLUTED')
            zlabel(app.UIAxes_4, 'Z')
            app.UIAxes_4.Color = [0.149 0.149 0.149];
            app.UIAxes_4.GridColor = [1 1 1];
            app.UIAxes_4.XGrid = 'on';
            app.UIAxes_4.YGrid = 'on';
            app.UIAxes_4.ZGrid = 'on';
            app.UIAxes_4.Position = [611 23 478 311];

            % Create STARTButton
            app.STARTButton = uibutton(app.UIFigure, 'push');
            app.STARTButton.ButtonPushedFcn = createCallbackFcn(app,
@STARTButtonPushed, true);
            app.STARTButton.BackgroundColor = [0 1 1];
            app.STARTButton.FontWeight = 'bold';
            app.STARTButton.Position = [296 719 100 23];
            app.STARTButton.Text = 'START';

            % Create STOPButton
            app.STOPButton = uibutton(app.UIFigure, 'push');
            app.STOPButton.ButtonPushedFcn = createCallbackFcn(app,
@STOPButtonPushed, true);
            app.STOPButton.BackgroundColor = [1 0 0];
            app.STOPButton.FontWeight = 'bold';
            app.STOPButton.Position = [730 719 100 23];
            app.STOPButton.Text = 'STOP';

            % Create HPCUTOFFFREQEditFieldLabel
            app.HPCUTOFFFREQEditFieldLabel = uilabel(app.UIFigure);
            app.HPCUTOFFFREQEditFieldLabel.HorizontalAlignment =
'right';
            app.HPCUTOFFFREQEditFieldLabel.FontWeight = 'bold';
            app.HPCUTOFFFREQEditFieldLabel.Position = [720 674 113
22];
            app.HPCUTOFFFREQEditFieldLabel.Text = 'HP CUT-OFF FREQ';
            % Create HPCUTOFFFREQEditField
            app.HPCUTOFFFREQEditField = uieditfield(app.UIFigure,
'numeric');
            app.HPCUTOFFFREQEditField.ValueChangedFcn =
createCallbackFcn(app, @HPCUTOFFFREQEditFieldValueChanged, true);
            app.HPCUTOFFFREQEditField.FontWeight = 'bold';
            app.HPCUTOFFFREQEditField.BackgroundColor = [0.8 0.8 0.8];
            app.HPCUTOFFFREQEditField.Position = [848 674 100 22];
            app.HPCUTOFFFREQEditField.Value = 30;

            % Create LPCUTOFFFREQEditFieldLabel
            app.LPCUTOFFFREQEditFieldLabel = uilabel(app.UIFigure);
            app.LPCUTOFFFREQEditFieldLabel.BackgroundColor = [0.502
0.502 0.502];
            app.LPCUTOFFFREQEditFieldLabel.HorizontalAlignment =
'right';
            app.LPCUTOFFFREQEditFieldLabel.FontWeight = 'bold';
            app.LPCUTOFFFREQEditFieldLabel.Position = [134 674 112
22];
            app.LPCUTOFFFREQEditFieldLabel.Text = 'LP CUT-OFF FREQ';

            % Create LPCUTOFFFREQEditField
            app.LPCUTOFFFREQEditField = uieditfield(app.UIFigure,
'numeric');
            app.LPCUTOFFFREQEditField.ValueChangedFcn =
createCallbackFcn(app, @LPCUTOFFFREQEditFieldValueChanged, true);
            app.LPCUTOFFFREQEditField.FontWeight = 'bold';
            app.LPCUTOFFFREQEditField.BackgroundColor = [0.8 0.8 0.8];
            app.LPCUTOFFFREQEditField.Position = [260 674 100 22];
            app.LPCUTOFFFREQEditField.Value = 200;

            % Create ORDEROFFILTEREditFieldLabel
            app.ORDEROFFILTEREditFieldLabel = uilabel(app.UIFigure);
            app.ORDEROFFILTEREditFieldLabel.HorizontalAlignment =
'right';
            app.ORDEROFFILTEREditFieldLabel.FontName = 'AvantGarde';
            app.ORDEROFFILTEREditFieldLabel.FontWeight = 'bold';
            app.ORDEROFFILTEREditFieldLabel.Position = [424 674 113
22];
            app.ORDEROFFILTEREditFieldLabel.Text = 'ORDER OF FILTER';

            % Create ORDEROFFILTEREditField
            app.ORDEROFFILTEREditField = uieditfield(app.UIFigure,
'numeric');
            app.ORDEROFFILTEREditField.ValueChangedFcn =
createCallbackFcn(app, @ORDEROFFILTEREditFieldValueChanged, true);
            app.ORDEROFFILTEREditField.BackgroundColor = [0.8 0.8
0.8];
            app.ORDEROFFILTEREditField.Position = [553 674 100 22];
            app.ORDEROFFILTEREditField.Value = 4;

            % Create LableLabel
            app.LableLabel = uilabel(app.UIFigure);
            app.LableLabel.FontName = 'AvantGarde';
            app.LableLabel.FontWeight = 'bold';
            app.LableLabel.Position = [544 550 37 22];
            app.LableLabel.Text = 'Lable';

            % Create BPMLabel
            app.BPMLabel = uilabel(app.UIFigure);
            app.BPMLabel.FontSize = 14;
            app.BPMLabel.FontWeight = 'bold';
            app.BPMLabel.Position = [540 588 45 22];
            app.BPMLabel.Text = 'BPM: ';

            % Create RPMLabel
            app.RPMLabel = uilabel(app.UIFigure);
            app.RPMLabel.FontSize = 14;
            app.RPMLabel.FontWeight = 'bold';
            app.RPMLabel.Position = [549 232 36 22];
            app.RPMLabel.Text = 'RPM';

            % Create Label2
            app.Label2 = uilabel(app.UIFigure);
            app.Label2.FontName = 'AvantGarde';
            app.Label2.FontWeight = 'bold';
            app.Label2.Position = [545 195 43 22];
            app.Label2.Text = 'Label2';

            % Create ELECTRONICSTETHOSCOPELabel
            app.ELECTRONICSTETHOSCOPELabel = uilabel(app.UIFigure);
            app.ELECTRONICSTETHOSCOPELabel.HorizontalAlignment =
'center';
            app.ELECTRONICSTETHOSCOPELabel.FontName = 'Bell MT';
            app.ELECTRONICSTETHOSCOPELabel.FontSize = 36;
            app.ELECTRONICSTETHOSCOPELabel.FontWeight = 'bold';
            app.ELECTRONICSTETHOSCOPELabel.Position = [295 741 534
47];
            app.ELECTRONICSTETHOSCOPELabel.Text = 'ELECTRONIC
STETHOSCOPE';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = GUI

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)
            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

3.2) GITHUB LINK
https://github.com/Kaage07/Electronic-stethoscope/tree/main

REFERENCE LIST

[1] Yu, F., Bilberg, A., & Voss, F. (2008, October). The development of an intelligent electronic stethoscope. In 2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications (pp. 612-617). IEEE.
[2] Rennoll, V., McLane, I., Emmanouilidou, D., West, J., & Elhilali, M. (2020). Electronic stethoscope filtering mimics the perceived sound characteristics of acoustic stethoscope. IEEE journal of biomedical and health informatics, 25(5), 1542-1549.
[3] Malik, B., Eya, N., Migdadi, H., Ngala, M. J., Abd-Alhameed, R. A., & Noras, J. M. (2017, September). Design and development of an electronic stethoscope. In 2017 Internet Technologies and Applications (ITA) (pp. 324-328). IEEE.
[4] Tran, M., Kuwana, A., & Kobayashi, H. (2020, March). Derivation of Loop Gain and Stability Test for Multiple Feedback Low Pass Filter Using Deboo Integrator. In The 8th IIAE Int. Conf. on Industrial Application Engineering.
[5]